

Localized Algorithm of Community Detection on Large-Scale Decentralized Social Networks

Pili Hu

Wing Cheong Lau

Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract—Despite the overwhelming success of the existing Social Networking Services (SNS), their centralized ownership and control have led to serious concerns in user privacy, censorship vulnerability and operational robustness of these services. To overcome these limitations, Distributed Social Networks (DSN) have recently been proposed and implemented. Under these new DSN architectures, no single party possesses the full knowledge of the entire social network. While this approach solves the above problems, the lack of global knowledge for the DSN nodes makes it much more challenging to support some common but critical SNS services like friends discovery and community detection. In this paper, we tackle the problem of community detection for a given user under the constraint of limited local topology information as imposed by common DSN architectures. By considering the Personalized Page Rank (PPR) approach as an ink spilling process, we justify its applicability for decentralized community detection using limited local topology information. Our proposed PPR-based solution has a wide range of applications such as friends recommendation, targeted advertisement, automated social relationship labeling and sybil defense. Using data collected from a large-scale SNS in practice, we demonstrate our adapted version of PPR can significantly outperform the basic PR as well as two other commonly used heuristics. The inclusion of a few manually labeled friends in the Escape Vector (EV) can boost the performance considerably (64.97% relative improvement in terms of Area Under the ROC Curve (AUC)).

I. INTRODUCTION

In the past few years, large-scale Social Networking Services (SNS) such as Facebook, Twitter and Renren have become a major part of people's daily life. Besides serving as a communication and information-sharing platform for their users, these Online Social Networks (OSN) also play a key role in friends discovery and community formation for users of common interests. Despite the overwhelming success of the existing SNS, the centralized ownership and control of these services have led to serious concerns in user privacy, censorship and operational robustness. Since the operator of an SNS has full knowledge of the profiles, social relationships and communication activities of their users, it is a high-value and obvious target for not only the typical attackers but also many totalitarian regimes which constantly seek to monitor and control information dissemination among their people.

To overcome the drawbacks of centralized OSNs, Decentralized Social Networks (DSN) such as Diaspora [25], Musubi [9] and OneSocialWeb [26] have recently been proposed and implemented. Under these DSN architectures, no single party possesses the full knowledge of the entire social network. For example, in the Diaspora network, a server (super-node) only has its own partial view of those users registered within it. It

is even more restrictive in the Musubi network in which every user only has visibility of his/her own relationships in form of a contact book; even if a user can consult his direct friends for their friend lists, his knowledge of the network is still constrained to 2 hops. Current efforts of Decentralized Social Network (DSN) are mainly on system prototyping. Although the design of many building blocks for DSN can be adapted from prior P2P system designs, the nature of DSN imposes further challenges: Due to privacy and trust concern, available information for a single node is limited. For example, in a BT swarm, nodes are willing to exchange content with complete strangers as long as they provide each other enough upload rate. In DSN, however, people only want to share contents (profile, status, blog, album, etc) within certain scope.

While the compartmentalized approach of the DSNs can enhance user privacy and make them less vulnerable to monitoring, censorship and sabotage attacks, the lack of global knowledge for the DSN nodes makes it much more difficult to support some common but critical services such as friend discovery/ recommendations and community detection. In particular, the DSN architecture invalidates many assumptions on data availability and global network topology awareness which are required by most existing community detection algorithms.

To address this challenge, this paper considers the following problem for the DSNs: Given a user and his/her partial knowledge of the social graph, can we predict which nodes are in the same *community* with him? The solution of this problem can support many fundamental services of DSN including friends recommendation, targeted advertisement, automated relationship labeling and sybil defense. In particular, we will focus on the design and performance evaluation of *localized* and *fully-decentralized* community detection algorithms for *large-scale* DSNs based on the Personalized Page Rank (PPR) approach:

- By “localized”, we mean that every node's knowledge of the network is limited to its local neighborhood topology¹. This is in contrast to other so-called localized approaches, e.g. the local graph partitioning algorithm in [4], for which global network topology is actually available even though the algorithm may choose not to fully explore or leverage such information.

¹While it has been shown that one can substantially improve community detection performance by incorporating information beyond mere topological data [21], [29], as an initial study of community detection on DSN, we consider only the topological data of the network for now.

- By “fully-decentralized”, we mean that every individual node within the DSN can execute our proposed community detection algorithm based on its locally acquired knowledge. Explicit coordination from other nodes is not needed. In contrast, many existing “distributed community detection” algorithms, e.g. those proposed in [13] and [28], do require exchange of information and collaboration among the network nodes.
- By “large-scale”, we mean that our algorithm is designed to be scalable for DSNs of sizes comparable to the existing top-tier OSNs. In fact, we evaluate our proposed algorithm in Section V-A using real data collected from the Renren social network in China (which already had 160 million registered users by 2011 [36]). More importantly, even if we only consider the 2-hop local neighborhood of a target Renren user, the size of the topological data is already very substantial: on average, each user has around 350 direct friends and 75,000 friends-of-friends. In fact, the size of the local topology of a single target user in our study is already as large as many medium-sized data sets used for evaluating many global community detection algorithms reported in the literature.

In short, this paper has made the following technical contributions:

- We formulate the new problem of community detection under the constraint of limited local topology awareness found in DSNs.
- We adapt the Personalized Page Rank (Personalized PageRank (PPR)) algorithm for community detection under limited topology information. In particular, by interpreting PPR as an ink spilling process, we justify its applicability for community detection for a target user.
- We investigate different design choices of the Escape Vector (EV) for PPR. Their impact on community detection performance are evaluated using real-world data sets collected from the large-scale Renren social network.

The rest of the paper is organized as follows. In Section II, we survey related works. In Section III, we formulate the community detection problem on 2-hop-only topology of the social network graph. In Section IV, we analyze the problem and propose our algorithm based on the PPR approach. In Section V, we evaluate the performance of our PPR proposal by comparing it with other commonly-deployed heuristics, and study the effect of different choices of PPR Escape Vectors in depth. We then conclude the this paper in Section VII.

II. RELATED WORK

A. Community Detection Algorithms

Community detection is a classical problem of partitioning a graph into multiple (possibly overlapping) subsets (communities) while satisfying the following conditions: intra-community linkage is dense and inter-community linkage is sparse. In the literature, this principle has been realized in many different ways including Modularity maximization,

Conductance minimization and Normalized Cut minimization [2]. So far, Modularity maximization [23] has been the most popular approach for community detection. However, since the resultant combinatorial optimization problem is NP-Hard, researchers have proposed many heuristics to obtain approximate solutions. Newman [24] proposed an eigen decomposition approach plus a local search algorithm. Agarwal [1] leveraged several mathematical programming techniques. Those works are the first batch that focus on detecting community based on only full topology information.

Since pure topology-based community detection algorithms are well developed by now, many researchers are trying to incorporate more side information to enhance performance of existing algorithms. In [29], Sachan et al combine social graph topology, interaction pattern, and topics to discover topically meaningful communities. In [21], Lin studies the scenario where node-level information is complete (like authors’ bag-of-words coordinates) but only part of the topology is observed. Note however that complete node-level information is still too hard to be acquired in the DSN scenario.

All of the above community detection algorithms are centralized ones requiring global information. Besides the data constraint, those global algorithms are usually computational intensive, which makes them inapplicable for large-scale OSNs. As such, our study of scalable and localized algorithms is important even under the centralized OSN settings.

It is worth to note that researchers have proposed some algorithms to tackle with large-scale graphs in centralized settings. [27] proposed a Label Propagation (LP) algorithm whose complexity for one iteration is linear of number of edges. LP initializes nodes with their unique labels and nodes switch to the majority label owned by their neighbours at each iteration. [19] conducted more experiments on the LP algorithm and proposed several improvements. Since our algorithm only requires local topology of each node, it is easy to be parallelized and scale to large graphs naturally under the centralized settings. We leave the combination of multiple runs of our localized algorithm to form a global community to future work.

B. Friend Recommendation Systems in Practice

We mentioned several applications of community detection algorithms on DSN in Section I. In the literature, Friend Recommendation is also associated with the problem of link prediction [20]. Although many sophisticated algorithms have been developed, they also suffer from their forbidding computational requirement like global community detection algorithms do. For large-scale OSN in practice, simple heuristics are believed to be in use. Veneta [30], for example, uses Common Neighbor to do friend discovery but their focus is on how to achieve this simple heuristic securely. While no detail information is available for the proprietary friend recommendation algorithms used by commercial OSN service providers, a recent post [35] by the Tencet research team noted the deficiency of using Common Neighbors heuristic

for community detection. It also expressed their concern of the computational burden of mature global optimization methods.

C. Other Topics to Be Assisted by Community Detection

Community detection output can assist many other topics. Detecting community in decentralized scenario can also help research in those topics. In this section, we briefly discuss a few of them.

To design the DTN routing algorithm, BubbleRap [14], Pan used the knowledge of community and centrality as heuristics.

It is shown that video views preserves good locality in terms of both geo-locations [7] and social network topology [31]. The key observation is that, many interest diffusion processes have some target communities. Community detection result can help to improve the performance of those social network assisted content distribution.

Social network assisted Sybil Defense also draws a lot of attention in recent years. It includes sybil identity detection and sybil community detection [32]. The latter one is the same problem as community detection. What differs is that the structural assumptions in Sybil Defense schemes are usually stronger and its application is more sensitive to TP or FP depending on the scenario. With locally detected communities, nodes can share information to form a trust region, thus excluding nodes who are highly susceptible to be sybil.

III. PROBLEM FORMULATION

A. Define Local Topology Using The Notion of Hops

We model the social network as a graph where each node corresponds to a person and each link corresponds to a relationship. For social networks in practice, relationships can be directed (like following on Twitter) or undirected (like being friends on Facebook). In this work, we focus on undirected relationship only.

Let's denote the undirected social graph as $G = (V, E)$, where V is the vertex (node) set and E is the edge (link) set. We also denote a node $v_i \in V$ using the shorthand notation i . Given a node i , we denote its 0-hop neighbour set as $N_i^{(0)} = \{i\}$. Iteratively, we can define the h -hop neighbour set as $N_i^{(h)} = \{j | k \in N_i^{(h-1)}, j \in V, (k, j) \in E\} \cup N_i^{(h-1)}$, where $h \geq 1$. The set of edges that can be observed within h -hop is $E_i^{(h)} = \{(j, k) | j \in N_i^{(h-1)}, k \in N_i^{(h)}, (j, k) \in E\}$.

Using the above notations, we can define the local topology information available to a user as follows: We call a normal user of SNS as "observer", denoted by o . How much information the observer can get depends on the application scenario. Given an observer o , the h -hop local topology is $G_o^{(h)} = (N_o^{(h)}, E_o^{(h)})$. To facilitate discussions, we define $n = |N_o^{(h)}|$ to be the number of nodes and $m = |E_o^{(h)}|$ to be the number of edges.

B. Illustration of 2-Hop Topology

Fig. 1 gives an illustration of $G_o^{(2)}$. In Fig. 1(a), the central biggest node (O) is the observer. Nodes denoted by "L1" are the direct friends of the observer, also referred to as the level 1 nodes (reachable within 1 hop). Similarly, nodes denoted by

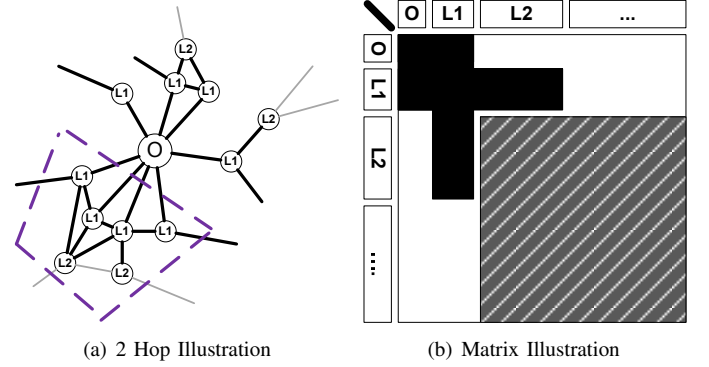


Fig. 1. 2 Hop Topology Illustration and Matrix Form

"L2" can be reached within 2 hops and are referred to as level 2 nodes. Dark thick edges show the links which are visible to the observer. Light thin edges show possibly existing links but the observer cannot see them within its 2 hop visibility. Dashed region shows a probable community since those nodes have dense linkage inside and sparse linkage outside. Fig. 1(b) illustrates the observer's knowledge of the adjacency matrix. Nodes are ordered and labeled using the same convention as the left panel. The black area is the part known to the observer. The white vacancy represents zeros in the adjacency matrix. The shadowed square is unknown, including the light thin edges in the left figure.

C. Justifications for a 2-Hop Formulation

As an initial study of community detection on DSN, we formulate the problem assuming only 2-hop topology information is available for the target user. The reasons are:

- **2-Hop is the smallest topology to study.** The 1-hop subgraph from an observer is a simple star, and there is no further topology information to utilize.
- **OSN diameter continues to shrink.** Dating back to 1929, Frigyes Karinthy proposed the concept of 6 degree(7 hops) separation [34]. Stanley Milgram conducted experiments in 1960 to show an average degree of 5.2 (6 hops) [33]. According to Facebook's study in the end of 2011 [33], the separation in the Facebook network was 5.28 hops in 2008 but became 4.74 hops in 2011.
- **Average node degree is very high.** Dunbar's Number [10] estimates the number of stable relationships of a typical person to be 150. The Renren network (further explained in Section V-A) is much better connected than that and the size of topology data associated with a local neighborhood would increase rapidly with the hop-count.
- **There are existing heuristics on only 2-hop topology.** Heuristics like common neighbors and Adamic/Adar score [2] (further explained in Section V-E) can be computed exactly within 2 hop and provide reasonable baseline for our study.
- **Trust concern prevents user from looking further away.** In the extreme case of DSN, users only have their own first hop topology (e.g. through Musubi's contact

book). The 2-hop topology can be obtained by requesting those direct friends and is reliable. However, asking a stranger for his buddy list may risk at polluting the raw data and resulting in meaningless detection result.

D. Community Detection As a Classification problem

Conventional community detection is formulated as a clustering problem. That is, given the full graph $G = (V, E)$, partition the vertex set into K subsets $\{C_1, C_2, \dots, C_K\}$, such that $\cap_{i=1}^K C_i = \emptyset$ and $\cup_{i=1}^K C_i = V$. The notion of community is that nodes within the same community have dense linkage and nodes from different communities have sparse linkage. To implement this concept mathematically, people proposed different quality measures $Q(\cdot)$ on a partitioning. Thus the clustering version of community detection problem becomes (consider the non-overlapping case)

$$\underset{\{C_1, C_2, \dots, C_K\}}{\text{maximize}} \quad Q(\{C_1, C_2, \dots, C_K\}) \quad (1)$$

$$s.t. \quad \cap_{i=1}^K C_i = \emptyset \quad (2)$$

$$s.t. \quad \cup_{i=1}^K C_i = V \quad (3)$$

Whether the “maximize” or “minimize” operator is used depends on the nature of the quality measure. One implementation of $Q(\cdot)$ is the Modularity proposed by Newman [23]:

$$Q = \frac{1}{2m} \sum_{k=1}^K \sum_{i,j \in C_k} (A_{ij} - \frac{d_i d_j}{2m}) \quad (4)$$

where A is the adjacency matrix of G , $d_i = \sum_j A_{ij}$ is the degree of node i , $m = |E|$ is the total number of edges. Modularity in essence measures how far the resulting partition is from a random graph and, the higher the better. There are other quality measures, e.g. Conductance. Interested readers can refer to [2] for more information.

The clustering formulation draws a lot of interest in the past. One reason is that the form is clean and highly amenable for theoretical studies. The other reason is that at the time when people started to do community detection, there were no successful large-scale OSNs. Researchers did not have ground-truth to validate whether a community partitioning is *correct*. So the clustering formulation and quality function evaluation became the mainstream approach in the research community.

With the wide acceptance of OSN in the recent years, labeled data become available. For instance, in the Renren social network graph, every node is associated with an institution name I_i , representing his/her university, high school, company, etc. Note that a person may have multiple institutions in reality. For the moment, we consider the social graph with only one institution name I_i for each node i . Details regarding multiple institution names are put in Section V-A.

Since the ground-truth is available, evaluating the community detection result against the crawled ground-truth is more reasonable. Mathematically speaking, given an observer o and its 2-hop local topology $G_o^{(2)}$, we want to determine $\forall i \in T_o$ whether $I_i = I_o$, where $T_o \in V$ is the set of test nodes. $T_o = N_o^{(2)}$ means that we evaluate the algorithm on

# of Samples	$\hat{L} = 1$	$\hat{L} = 0$
$L = 1$	a	b
$L = 0$	c	d

Fig. 2. Confusion Matrix

all nodes within 2 hops, which fits the application of friend recommendation. $T_o = N_o^{(1)}$ means that we only evaluate the algorithm on only level 1 nodes, which fits the application of automated friend categorization. Both choices of T_o will be evaluated in Section V. We then cast this decision problem as a binary-classification problem, i.e. assign a label L_i (the ground-truth) to each node:

$$L_i = \begin{cases} 1 & I_i = I_o \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The algorithm should take the 2-hop topology as input and output predicted labels $\hat{L}_i \in \{1, 0\}$.

The confusion matrix of classification result is shown in Fig. 2. Using the notation in the confusion matrix, we have multiple standard ways to evaluate the quality of predicted labels \hat{L}_i , such as accuracy:

$$\text{Accuracy} = (a + d)/(a + b + c + d) \quad (6)$$

true positive rate

$$\text{TPR} = a/(a + b) \quad (7)$$

and false positive rate

$$\text{FPR} = c/(c + d) \quad (8)$$

In Section V, we will further analyze the shortcomings of the above evaluation methods and propose to use Receiver Operating Characteristics (ROC) and Area Under the ROC Curve (AUC) to evaluate key steps of the algorithm in depth.

IV. ANALYSIS AND ALGORITHM DESIGN

In this section, we analyze the problem and propose a two-stage algorithm framework first. Then we shift our focus on the first stage, i.e. finding a reasonably good ranking function of all nodes in T_o . Among all kinds of possibilities, we investigate the variations of PageRank (PR) and Personalized PageRank (PPR). We show PR is not informative in our setting and justifies the potential of PPR using an “ink spilling” process. In the last part of this section, we analyze the complexity of PPR computation using both conventional matrix multiplication method and ink spilling algorithm.

A. Two-stage Classification Framework

There are many ways to construct good classifiers. In this work, we investigate a simple framework as shown in Algorithm 1. In the first stage, we compute a ranking function $f(i)$, which assigns scores to all the nodes concerned. This score reflects how likely one node is positive but it does not necessarily be strict probability distribution on T_o . In the later discussion, we also refer to $f(i)$ as “feature”, “heuristics” and “ranking” interchangeably. We can consider the two metrics

Algorithm 1 Community Classification Framework

Input: Observer o , 2-hop topology $G_o^{(2)}$, test set T_o

Output: Predicted labels: $\hat{L}_i, \forall i \in T_o$

- 1: Compute a ranking function $f(i), \forall i \in T_o$
- 2: Compute a proper threshold $R \leftarrow g(o, G_o, f)$
- 3: Binary classification by thresholding:

$$\hat{L}_i \leftarrow \begin{cases} 1 & f(i) \geq R \\ 0 & f(i) < R \end{cases}$$

defined in Eq. 14 and Eq. 15 as two examples of $f(i)$. In the second stage, we compute a proper threshold and cut the nodes into positive and negative sets according to this threshold.

Note that our framework itself may not be optimal. For example, we can compute more than one feature, and train a probably better classifier. As a pilot study of the potential of only 2-hop topology, we, however, are more interested in what performance a single ranking function can achieve in this setting. The investigation on the use of multi-dimensional ranking function is left as future work.

B. PageRank and the Limitation of Its Basic Form

We denote the adjacency matrix of a graph by A . The element of degree vector \vec{d} is defined as $\vec{d}_i = \sum_j A_{ij}$. The degree matrix is defined as $D = \text{diag}(\vec{d}_1, \vec{d}_2, \dots, \vec{d}_n)$. Left multiplying A by the inverse degree matrix, we get the normalized adjacency matrix (also called the walk matrix): $W = D^{-1}A$. The PR vector \vec{v} is obtained by solving the following fixed-point equation:

$$\vec{v} = W^T \vec{v} \quad (9)$$

The solution of \vec{v} can also be interpreted as the stationary probability distribution of a Random Walker (RW) on the graph. In each step, the RW uniformly picks an adjacent edge and walks to the other side of the edge. We consider one situation that an RW gets bored and restarts from a random nodes uniformly (called “escaping”). The escaping probability is denoted by $(1 - \alpha)$ and the corresponding stationary distribution can be solved through the following equation:

$$\vec{v} = \alpha W^T \vec{v} + (1 - \alpha) \frac{\vec{1}}{\|\vec{1}\|_1} \quad (10)$$

where $\vec{1}$ is the all ones vector, and $0 < \alpha < 1$.

The PR algorithm is very successful in web ranking context [6]. It is shown to satisfy a set of ranking axioms [3], that fits the web ranking setting very well. The RW interpretation also fits our problem. Since our partial topology is biased towards o , nodes with higher proximity with o should be visited more frequently by an RW. If one node has higher proximity with the observer, it is more likely to be in the same community. This intuition is the same as simple heuristics like Common Neighbour and Adamic/Adar Score [2].

The initial version of PR (Eq. 9) is not informative. This is because on an undirected and connected graph, the stationary

distribution satisfies $\vec{v} \propto \vec{d}$. However, higher degree does not necessarily mean higher proximity with the observer. The modified version of PR (Eq. 10) re-shares the score among nodes, and results in a different ranking. On one extreme, when $\alpha \rightarrow 0$, \vec{v} is a uniform distribution. On the other extreme, when $\alpha \rightarrow 1$, \vec{v} is the degree-proportional distribution. This version is more informative than the initial version in Eq. 9 and we will evaluate how it performs in Section V.

C. Personalized PageRank

Personalized PageRank (PPR), also known as topic sensitive pagerank [12] is a generalization of Eq. 10:

$$\vec{v} = \alpha W^T \vec{v} + (1 - \alpha) \frac{\vec{b}}{\|\vec{b}\|_1} \quad (11)$$

where \vec{b} is the personalized Escape Vector (EV), and $\|\vec{b}\|_1 = \sum_i |\vec{b}_i|$. To simplify the notation, we denote $\vec{\beta} = \vec{b} / \|\vec{b}\|_1$ as the normalized version of \vec{b} . Instead of filling all ones in \vec{b} , we can make the escaping probability biased towards a set of nodes. The intuition is that the RW will restart from some nodes with particular interest instead of uniformly pick a random node. In the stationary distribution, nodes close to those restarting nodes will have higher probability. If we know some positive nodes beforehand, putting them in PPR’s EV will help to rank other positive nodes higher.

D. The Ink Spilling Interpretation of PPR

Besides the random walk view, PPR also has a nice interpretation using ink spilling process [16]–[18]. To see this, we reorganize Eq. 11 as $(I - \alpha W^T) \vec{v} = (1 - \alpha) \vec{\beta}$. The eigen value of $W^T = AD^{-1}$ is in the range $[-1, 1]$, so the eigen value of $(I - \alpha W^T)$ is in the range $[1 - \alpha, 1 + \alpha]$. Then $(I - \alpha W^T)$ is invertible and we have $\vec{v} = (I - \alpha W^T)^{-1} (1 - \alpha) \vec{\beta}$. Since the eigen value of αW^T is in the range $[-\alpha, \alpha]$, we can perform the expansion $(I - X)^{-1} = I + X + X^2 + X^3 \dots$ by taking $X = \alpha W^T$ in the above equation:

$$\vec{v} = (1 - \alpha) \sum_{t=0}^{\infty} (\alpha W^T)^t \vec{\beta}. \quad (12)$$

Eq. 12 has an intuitive interpretation using an ink spilling process:

- 1) Every node i is initialized with $\vec{\beta}_i$ amount of ink.
- 2) At each step, $(1 - \alpha)$ of the ink dries at every node.
- 3) The rest α portion wet ink is passed to neighbors using the weighting of W^T .
- 4) The process repeats until all ink dries.

Based on this ink spilling interpretation, we can put some user labeled positive nodes in EV. Since the positive nodes form a community which is well connected inside and have sparse connection with the rest network, positive nodes should generally have more ink than negative nodes. If more positive nodes are put into EV, PPR should be more capable to distinguish between positive and negative nodes. We will numerically investigate the sensitivity of PPR to the size of restarting set in Section V.

Algorithm 2 Ink Spilling Algorithm

Input: Walk matrix: W
Input: Escape vector: \vec{b}
Output: ϵ -approximation: \vec{v}_ϵ

```

1:  $\vec{r} \leftarrow \vec{\beta}$ 
2: while  $\exists i, \text{s.t. } \vec{r}(i) > \epsilon \vec{d}(i)$  do
3:    $\vec{v}_\epsilon(i) \leftarrow \vec{v}_\epsilon(i) + (1 - \alpha)\vec{r}(i)$ 
4:    $\forall j \in N_i^{(1)}, \vec{r}(j) \leftarrow \vec{r}(j) + \alpha W_{ij}\vec{r}(i)$ 
5: end while
```

E. Complexity of PPR

Computing PPR can be very efficient. We analyze two methods: Matrix Multiplication and Ink Spilling.

1) *Matrix Multiplication*: Based on Eq. 11, we can derive the straightforward iteration:

$$\vec{v}^{(t)} = \alpha W^T \vec{v}^{(t-1)} + (1 - \alpha)\vec{\beta} = P\vec{v}^{(t-1)} \quad (13)$$

where $P = (\alpha W^T + (1 - \alpha)\vec{\beta}\vec{1}^T)$. The convergence rate of Eq. 13 depends on the eigen gap of P . It can be shown that $\lambda_2(P) \leq \alpha$ [15], which provides a constant gap irrelevant of n . Then $\Theta(\log n)$ iterations are needed to approximate the stationary distribution with error bounded by $\|\vec{v}^{(t)} - \vec{v}^{(\infty)}\| = O(\frac{1}{n})$ [18]. Basic matrix-vector multiplication costs $O(n^2)$ time. In our implementation, we deal with the two parts of P separately. They cost $O(m)$ and $O(n)$ time, respectively. The total complexity for a fixed α is then $O((m + n) \log n)$. As is shown before, on the 2-hop topology, m is on the order of n , so the computation is tractable for every single user on commodity desktop computers.

2) *Ink Spilling*: Note that the ink spilling process can be performed in an asynchronous manner [18]. We formalize the asynchronous version in Algorithm 2. Denote the level of wet ink at node i by $\vec{r}(i)$. At each step, we pick an arbitrary node satisfying the condition $\vec{r}(i) > \epsilon \vec{d}(i)$, dry $(1 - \alpha)$ portion of wet ink, and share the remaining ink among its direct neighbors. If no such node exists, i.e. $\vec{r}(i) \leq \epsilon \vec{d}(i), \forall i$, we call the current \vec{v}_ϵ an ϵ -approximation of PPR. The termination condition implies that there is only a small fraction of wet ink at all nodes.

We denote the node selected at step t by v_t . From the iteration invariance, we know at least $\epsilon \vec{d}(v_t)$ ink is wet at this node. Since the initial ink level is $\sum_i \beta_i = 1$, we have $\sum_t (1 - \alpha)\epsilon \vec{d}(v_t) \leq 1$. That is $\sum_t \vec{d}(v_t) \leq 1/((1 - \alpha)\epsilon)$. In each step, it takes constant time to dry a portion of ink for the current node and $\vec{d}(v_t)$ time to distribute wet ink to neighbors. So the complexity of ink-spilling algorithm is $\sum_t \vec{d}(v_t) = O(\frac{1}{(1 - \alpha)\epsilon})$ [18]. To reach the same level of approximation as matrix multiplication, i.e. $\sum_i \epsilon \vec{d}(i) = \Theta(\frac{1}{n})$, we have $\epsilon = \Theta(\frac{1}{mn})$. As α is constant for a given graph size, the overall complexity is $O(mn)$. Note that in the real application, performing excellent approximation is not necessary. So we can set ϵ to a larger value than $\Theta(\frac{1}{mn})$. We also note that this analysis is not tight and in practice ink spilling algorithm can benefit from the sparse structures of many graphs.

V. PERFORMANCE EVALUATION

We have evaluated our proposed algorithms using data sets crawled from Renren, currently the largest OSN in China.

A. Data Set

By default, a Renren user can only view the buddy list of their direct friends. This setting incidentally restricts the network perspective of an individual Renren user to his/her 2-hop neighborhood, matching exactly the scenario we considered in Section III. We have developed a dedicated crawler to collect 2 hop buddy list information (friends and friends of friends) from any given observer. Eight volunteers have helped to run the crawler and contributed their own data. All the volunteers are active undergraduate Renren users for classes 2006 to 2008. Our study is based on the separately anonymized version of their data. As our formulation only relies on 2-hop topology of any target user, we actually have collected eight independent data sets for evaluation. Since the results for these eight data sets are qualitatively the same, in what follows, we only present the quantitative evaluation results for one observer's data set (denoted by $G_o^{(2)}$) for brevity.

Manual examination of the data sets has reviewed that one's largest and also latest community always (in our data sets) correspond to his/her undergraduate university's community. In this initial evaluation, we are interested in the ability of our PPR approach in "re-discovering" this community based on 2-hop topology. For observers who have changed their institution name after graduation, we manually set his/her institution name to the undergraduate university. For any other nodes, we stick to the default institution name shown on the buddy list. All nodes that have the same institution name as the observer are regarded as positive nodes, and the rest are regarded as negative nodes. This preprocessing causes certain noise of the ground-truth but this is the best we can do for it's impossible for us to get all institution names of a person by default.

B. Evaluation Methodology

In Section III, we listed some conventional evaluation criterion for classification problem. However, simple evaluation like accuracy can be misleading. For instance, in our experiment, observer o has 88238 nodes in his 2-hop neighborhood ($L1 + L2$ friends). 9345 of them are positive nodes and the ratio of is 0.1059. That means, the most naive algorithm which declares all nodes to be negative can reach an accuracy (Eq. 6) of 0.8941. This result is not of practical interest. Instead of accuracy, we look at True Positive Rate (TPR) and False Positive Rate (FPR) separately.

In our two-stage algorithm framework, we first calculate a ranking of all nodes in T_o and then set a proper threshold R to cut the set into positive and negative set. By varying the threshold, we can obtain a series of TPR and FPR. Plotting all those points on a 2-D graph, we can get the Receiver Operating Characteristics (ROC) curve [11]. If we rank nodes randomly, the ROC curve is approximately a straight line connecting (0,0) and (1,1). The higher the ROC curve is above

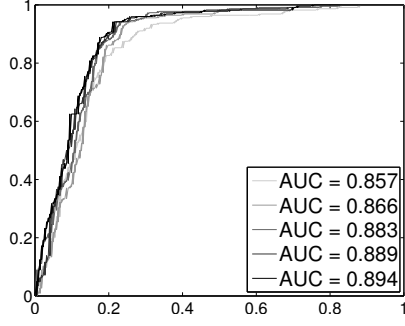


Fig. 3. ROC Curves with AUC Increases from Light to Dark

the diagonal line, the more capable a ranking is to distinguish between positive and negative nodes.

Although ROC is very informative for human, it is hard to automatically compare many ROC curves directly. Instead, the Area Under the ROC Curve (AUC) [11] is a better summary metric. If the AUC is larger, the ROC curve is generally higher. AUC is equivalent to the probability that an algorithm will rank a randomly chosen positive node higher than a randomly chosen negative node [11].

C. Validation of AUC as An Effective Metric

Although AUC reflects the expected performance of a classifier, it does not mean higher AUC curves will dominate lower AUC curves everywhere. In this section, we experimentally show that AUC is an effective metric.

In this experiment, we configure EV of PPR randomly. We run 10 instances of simulations and plot the ROC curves in Fig. 3. The test set T_o is set to L1 nodes ($T_o = N_o^{(1)}$). To make the plot uncluttered, only 5 of the curves are plotted. It is shown that larger AUC in general implies higher ROC curve. This is partly because those curves come from the same family, i.e. PPR value under different settings.

For the rest of our evaluation, we will focus on comparing the AUC of different approaches.

D. Leverage ROC to Get Classification Result

Getting a good ROC curve is just an intermediate step towards the final application. Before we extensively evaluate AUC of different approaches under different settings, we briefly note two methods to leverage the ROC curve to get classification result.

Suppose we get one ROC curve in Fig. 3. One attainable ROC point (0.1832, 0.8991) can be read from the plot. In $N_o^{(1)}$, there are 224 positive nodes and 222 negative nodes. This means that an algorithm can return $0.1832 \times 222 = 40.670$ negative nodes and $0.8991 \times 224 = 201.40$ positive nodes with proper threshold. The problem now becomes how to target this ROC point without knowing the ground-truth. Here are two choices:

- With prior knowledge of the portion of positive nodes (obtained through statistics over the network), we can target an ROC point in the following way. For example,

TABLE I
AUC OF FOUR DIFFERENT APPROACHES

Common	0.7415	Adamic/Adar	0.7428
PR	0.7024	PPR	0.8339

suppose we find (0.19, 0.9) is a reasonable ROC point to target. For a typical undergraduate user, he/she usually have half of the friends come from the same university. Then we vary threshold R until the number of predicted positive nodes reaches $0.19 \times 0.5 \times |N_o^{(1)}| + 0.9 \times 0.5 \times |N_o^{(1)}| = 0.545|N_o^{(1)}|$.

- Another method is to look at PPR values directly and detect possibly existing sharp drops. Andersen [5] studied the local graph partitioning problem and proposed to cut the graph by detecting sharp drops of the PPR vector. We already note the difference between Local Graph Partitioning and the “localized” setting of our problem in Section I. Whether similar approaches are applicable to the current problem is left to future work.

E. Comparison of Four Approaches

We first compare our PR and PPR proposal to two local heuristics:

- Common Neighbour (Eq. 14) simply computes the size of intersection between one node’s neighborhood and the observer’s neighborhood. The intuition is that the more friends the two nodes have in common, the more likely they are in the same community.

$$\text{Common}(i, j) = |N_i^{(1)} \cap N_j^{(1)}| \quad (14)$$

- Adamic/Adar score (Eq. 15) is an intuitive improvement of Common Neighbour. It biases towards lower-degree nodes. The reason is that higher degree nodes are more popular and thus provide little information if two nodes both connect to it. On the contrary, if two nodes both connect to a low-degree node, they have a larger chance of being in the same community.

$$\text{Adamic/Adar}(i, j) = \sum_{k \in N_i^{(1)} \cap N_j^{(1)}} \frac{1}{\log |N_k^{(1)}|} \quad (15)$$

$f(i) = \text{Common}(o, i)$ and $f(i) = \text{Adamic/Adar}(o, i)$ are computed to rank node i , respectively. The two simple heuristics can be computed exactly within 2-hop topology and are thus the right baseline to compare. For many other algorithms, their original settings are beyond 2-hop. Running those algorithms on our data set naturally violates their assumptions so the performance is not guaranteed.

In this experiment, the EV \vec{b} of PPR is simply set to 3 highest degree positive nodes plus the observer o . We defer the exploration of how the choice of \vec{b} would influence the performance to following sections. For the basic version of PR, the EV is set to an all-one’s vector. Both PPR and PR use a transition ratio of $\alpha = 0.9$. The results of the four approaches are evaluated on all nodes within 2-hop, i.e. $T_o = N_o^{(2)}$.

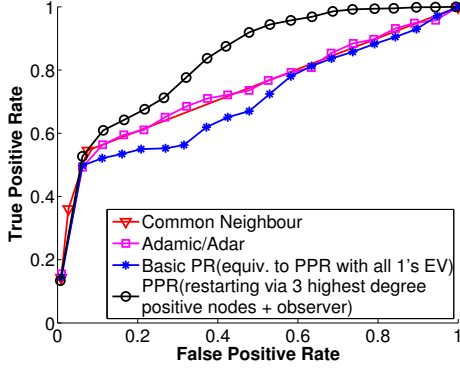


Fig. 4. ROC Curve of Four Different Approaches

Table I shows the AUC of the four approaches and Fig. 4 shows their ROC curves. We can see that Common Neighbours and Adamic/Adar perform equally well. In the low FPR region (e.g. $\text{FPR} < 0.1$), the four approaches do not have much difference. In the higher FPR region, PPR is obviously more effective than the others while pure PR even does worse than the two simple heuristics. Note that the AUC is at least 0.5 for meaningful approaches, PPR actually improves PR by 64.97% relatively. This result aligns with our analysis in Section IV.

Remark 1. A few labels help a lot. If the user can manually label a few positive nodes, the PPR algorithm can leverage this information to yield significant improvement.

F. Evaluation on All Nodes within 2-Hop

We already observed that a slight modification of EV \vec{b} can make a considerable difference in the ROC curve, whereas simple-minded all one's \vec{b} performs worse than the two local heuristics. In the previous proof-of-concept experiment, we put 3 highest degree positive nodes and the observer in the EV. In this section, we explore how AUC varies with different choices of \vec{b} .

Among different alternatives to fill in \vec{b} , we choose the unweighted version of \vec{b} for simplicity. That is, we determine a restarting set of nodes, denoted by $S \subset N_o^{(2)}$, and fill in \vec{b} as follows:

$$\vec{b}_i = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases} \quad (16)$$

1) *Random Positive Nodes as Restarting Set:* In this experiment, some random positive nodes are put into S . For each size of S , we repeat the experiment for 50 rounds to absorb randomness. Fig. 5(a) is the scatter plot of 5000 simulations with $|S|$ varying from 1 to 100. Fig. 5(b) plots the mean AUC of each $|S|$. It is clear that more pre-known positive nodes help to improve the performance of PPR on our 2-hop community detection settings but there is only a little benefit (0.03 increase of AUC from $|S| = 1$ to $|S| = 100$, i.e. an 8.8% relative improvement). Besides the increase of mean, larger $|S|$ results in smaller variance, which is also appreciated in real applications. Manual labeling causes a cost for the users, so they need to find a trade-off between performance and cost.

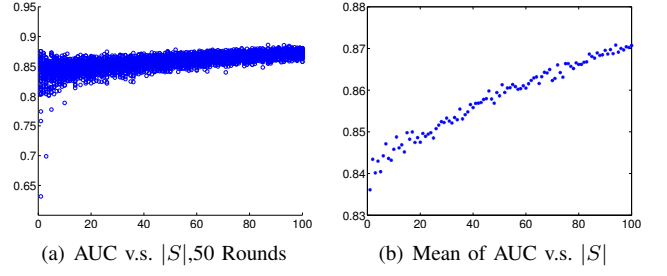


Fig. 5. Random S , AUC Evaluated on $N_o^{(2)}$

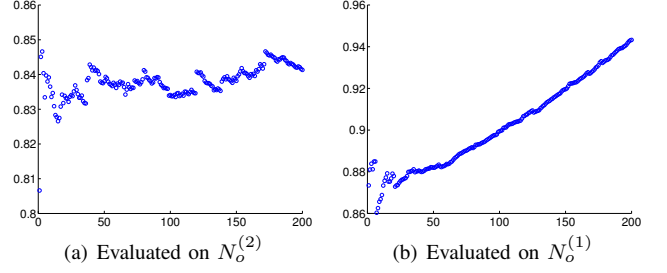


Fig. 6. AUC v.s. Number of High Degree Nodes

2) *High-Degree Positive Nodes as Restarting Set:* In this experiment, the x highest degree positive nodes are put into S . We vary x from 1 to 200. Fig. 6(a) shows the scatter plot of AUC v.s. x . There is no obvious pattern in this plot and high-degree heuristic generally performs worse than random positive node heuristic when $|S|$ is fixed. As a practical note, for the application on $N_o^{(2)}$ we suggest the user randomly label a few positive nodes.

Remark 2. More labels help a little. In the random setting, putting more nodes in the EV results in minor linear benefit. In the high degree setting, more nodes does not improve the performance.

G. Evaluation on Only Level-1 Nodes

In this section, we repeat the same experiment in the previous section but evaluate the result on $N_o^{(1)}$, i.e. only L1 friends. The detection result on $N_o^{(1)}$ cannot be used to do friend recommendation, for those nodes are already connected to observer. Nevertheless, it is still meaningful for automated relationship categorization.

1) *Random Positive Nodes as Restarting Set:* We first choose the random S , and plot two similar graphs as before in Fig. 7. It shows that the mean of AUC does not increase with $|S|$ but the variance decreases with $|S|$. Differences in Fig. 7(b) are mainly due to randomness.

2) *High-Degree Positive Nodes as Restarting Set:* Next we evaluate high-degree heuristic and plot the result in Fig. 6(b). After a fluctuation period, the AUC goes up steadily with the increase of $|S|$. Note that we are evaluating PPR on L1 nodes in this experiment. On the 2-hop topology seen by our observer, the high degree positive nodes are also mostly L1 nodes. Consider the ink spilling process. More L1 positive

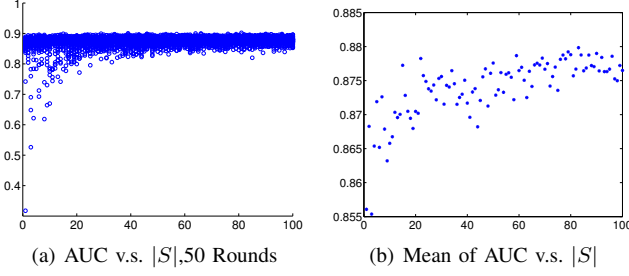


Fig. 7. Random S , AUC Evaluated on $N_o^{(1)}$

nodes in S means they have more initial ink. At least, nodes in S are guaranteed to have higher rank. The result of Fig. 6(b) is thus intuitive. Note that the curve in Fig. 6(b) waits until $|S| \approx 60$ to reach the linear increasing part. Labeling 60 highest degree positive nodes is a prohibitive workload for a normal user and is thus unrealistic in practice.

Comparing Fig. 6(b) with Fig. 7, the high degree positive heuristic generally performs better than random node heuristic. As a practical note, for the application on $N_o^{(1)}$, we suggest labeling several high degree positive nodes. High degree nodes in a community are not only easy to recognize, but also induce a deterministic construction of EV.

Remark 3. $G_o^{(h)}$ is the minimum for the application on $N_o^{(h-1)}$. Although PPR shows significant improvement to simple heuristics, the result on all 2-hop nodes is not good enough for practical use. This is shown by the ROC curves in Fig. 3 and Fig. 4. The evaluation on only level 1 nodes is promising, and we can find good ROC point to target in real applications.

H. Convergence Behavior of Two PPR Algorithms

In this section, we evaluate the runtime behavior of the two candidate realizations of PPR: Matrix Multiplication and Ink Spilling. The data set is from the same observer as above sections. To show the convergence behavior, we fix $\alpha = 0.9$ and set the EV using the 5 highest degree positive nodes. By varying ϵ for the two algorithms, we obtain the convergence rate and error. All computations are done on a laptop with 2.00 GHz CPU.

Note that the analysis of matrix multiplication algorithm in Section IV-E does not depend on the choice of vector norm. We use $\|\cdot\|_1$ in our implementation for both simplicity and numerical stability. Denote the result of algorithm “algo” with precision control parameter ϵ by $\vec{v}^{(\text{algo})}_\epsilon$. We choose $\vec{v}^{(\text{bm})} = \vec{v}^{(\text{matrix})}_{\epsilon=10^{-10}}$ as the benchmark. We record the difference $\|\vec{v}_\epsilon - \vec{v}^{(\text{bm})}\|_1$ and execution time for each ϵ . Fig. 8 plots the convergence behaviour of the two algorithms.

The x axis ($-\log_{10} \epsilon$) in Fig. 8 is in log-scale. We observe a linear part in both plots. This means the two algorithms both converges exponentially fast. In around 2 seconds, both of the algorithms can converge to a ranking function with $\|\vec{v}_\epsilon - \vec{v}^{(\text{bm})}\|_1 < 10^{-6}$, which is more than enough for most practical purposes. The flat part of the curve in Fig. 8(b) also

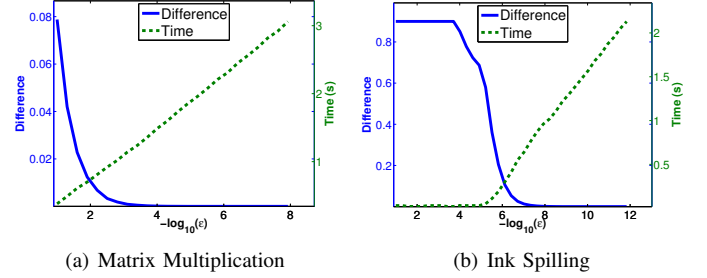


Fig. 8. Convergence Behaviour of Two Algorithms

indicates that more stringent ϵ should be set in order for the Ink Spilling approach to converge. This is because the definition of ϵ -approximation in the Ink Spilling algorithm involves node degree. For example, suppose we have a local topology with $|E(N_o^{(2)})| \approx 10^5$, at least $\epsilon \approx 10^6$ is required to make the total error upper bounded by 0.1 when the algorithm converges.

VI. DISCUSSIONS AND FUTURE WORKS

The extensive simulations above show that PPR is an effective and flexible proximity measure. With appropriate parameters, the PPR score can be used as a reasonable ranking function for our binary classification framework. We have also remarked that the application of PPR on L1 nodes are more appropriate for real application. There are good ROC points to target and we mentioned some methods, e.g. using prior distribution or detecting sharp drop, to target them. We illustrated the use of prior distribution. However, it is not trivial to set proper threshold in order to get final detection result. We leave this as a future work.

We have also remarked that 2-hop topology is not enough for detecting community of all 2-hop nodes (L1 + L2). Meanwhile, 2-hop topology is promising for community detection of L1 nodes. If we constrain the evaluation nodes to L1 nodes only, our formulation has become node classification on an ego-network. In the preparation of this manuscript, we found other similar works to tackle with node classification locally on an ego-network. In [22], the authors developed a probabilistic model that leverages both topology information and node-level information to discover social circles of an ego. DEMON [8] proposed to solve the centralized community detection of large graphs by first detecting communities locally and then combine them globally. It first uses Label Propagation [27] to detect community on ego-network of all nodes and then combine locally detected communities by thresholding their intersection. Our observations in the experiments and those works show that ego-network is a promising direction to go. Stepping back to our original problem, we see that AUC of PPR on L2 still has some improvements to basic heuristics. Although there are no good ROC points for direct application, the benefit brought by PPR may be useful in some extensions of the current problem.

One possible extension of the current problem is to allow cooperation among direct friends or a small number of collaborating observers in the network. Another direction is the

development of privacy preserving protocols for DSN which can allow users to help each other to improve their personal community detection without requiring excessive trust.

It is worth to note that overlapping community detection has also drawn a lot of interest in recent years. Our PPR proposal is readily available for overlapping community detection. In essence, PPR reveals the proximity of nodes to a set of seeds. The observer can provide different sets of seeds so as to reveal different communities. The detecting result can be overlapping.

In the simulation, it seems that we used more information than topology and people concerns about privacy of the data. We briefly discuss the data source as follows: 1) 2-hop topology is natural under most settings, e.g. on many OSN's, users can check their friends' buddy lists; 2) in the EV allocation, the observer manually label some positive nodes (or nodes he/she believes to be positive), thus requiring no collaboration from others; 3) in the high degree heuristic, degree information can be calculated directly on 2-hop topology (however the degree of L2 node is under estimated). Note that we can not know all node labels first and then sample random positive nodes or high degree positive nodes. In practice, user just identifies a set of positive nodes regardless of other node properties. This can be implemented and the effect should be similar to random node heuristic.

VII. CONCLUSION

In this paper, we have studied the community detection problem of any given user under the constraint of limited topology information imposed by emerging DSN's. In particular, we consider the scenario where only 2-hop topology information is available to the given target SNS user. Instead of following the clustering approach taken by most traditional works of community detection, our formulation yields a binary classification problem which can be evaluated against ground-truth data collected from real-world OSNs. We then establish a two-stage framework for this problem and transform our problem to the finding of a good ranking function of nodes in the test set. In particular, we adapt the notion of PPR for this purpose and justify its applicability on community detection via the Ink Spilling interpretation of PPR.

Our evaluation using real-world OSN data shows that even with the topology as small as 2-hop we can "discover" a target user's community. Our proposed PPR approach significantly outperforms the basic version of PR and two other heuristics believed to be commonly used by large-scale OSNs in practice. Our study shows that manually labeling just a few positive nodes for PPR can boost the performance.

REFERENCES

- [1] G. Agarwal and D. Kempe, "Modularity-maximizing graph communities via mathematical programming," *The European Physical Journal B-Condensed Matter and Complex Systems*, 2008.
- [2] C. Aggarwal, *Social network data analytics*. Springer-Verlag NY, 2011.
- [3] A. Altman, M. Tennenholtz, "The PageRank axioms," in *Procs. of the 6-th ACM conference on Electronic Commerce*, 2005.
- [4] R. Andersen, F. Chung, K. Lang, "Local graph partitioning using pagerank vectors," *FOCS*, 2006.
- [5] R. Andersen, F. Chung, "Detecting sharp drops in pagerank and a simplified local partitioning algorithm," *Theory and Apps. of Models of Computation*, 2007.
- [6] S. Brin, L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, 1998.
- [7] A. Brodersen, S. Scellato, and M. Wattenhofer, "Youtube around the world: Geographic popularity of videos," in *WWW*, 2012.
- [8] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: a local-first discovery method for overlapping communities," *KDD*, 2012.
- [9] B. Dodson et al, "Musubi: disintermediated interactive social feeds for mobile devices," *WWW2012*
- [10] R. Dunbar, "You've got to have (150) friends," *The New York Times, The Opinion Pages*, 2010.
- [11] T. Fawcett, "An Intro. to ROC analysis," *Pattern Recog. letters*, 2006.
- [12] T. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search," *IEEE Trans. on Knowledge and Data Engineering*, 2003.
- [13] P. Hui et al, "Distributed community detection in delay tolerant networks," in *Procs. of 2nd ACM/IEEE International workshop on Mobility in the evolving internet architecture*, 2007.
- [14] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008.
- [15] S. Kamvar, *Numerical Algorithms for Personalized Search in Self-organizing Information Networks*. Princeton Univ Press, 2010.
- [16] P. Berkhin, "Bookmark-coloring algorithm for personalized pagerank computing," *Internet Mathematics*, 2006.
- [17] D. Spielman, "Spectral graph theory lecture notes," 2009. [Online]. Available: <http://www.cs.yale.edu/homes/spielman/561/>
- [18] L. C. Lau, "Spectral algorithm lecture notes," 2012. [Online]. Available: <http://www.cse.cuhk.edu.hk/~chi/csc5160/index.html>
- [19] I. Leung, P. Hui, P. Lio, and J. Crowcroft, "Towards real-time community detection in large networks," *Physical Review E*, 2009.
- [20] D. Liben-Nowell, J. Kleinberg, "The link-prediction problem for social networks," *Journal of the Amer. soc. for Information Science & Technology*, 2007.
- [21] W. Lin et al, "Community detection in incomplete information networks," *WWW2012*.
- [22] J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *Neural Information Processing Systems*, 2012.
- [23] M. E. J. Newman, M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, Feb 2004.
- [24] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, 2006.
- [25] "Diaspora." [Online]. Available: <http://diasporaproject.org/>
- [26] "Onesocialweb," 2012. [Online]. Available: <http://onesocialweb.org/>
- [27] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, 2007.
- [28] L. Ramaswamy, B. Gedik, and L. Liu, "A distributed approach to node clustering in decentralized peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, 2005.
- [29] M. Sachan et al, "Using content and interactions for discovering communities in social networks," *WWW2012*.
- [30] M. Von Arb et al, "Veneta: Serverless friend-of-friend detection in mobile social networking," *IEEE WIMOB'08*.
- [31] Z. Wang, L. Sun, C. Wu, and S. Yang, "Guiding internet-scale video service deployment using microblog-based prediction," in *IEEE INFOCOM*, 2012.
- [32] H. Yu, "Sybil defenses via social networks: a tutorial and survey," *ACM SIGACT News*, 2011.
- [33] Facebook, 2011, Anatomy of Facebook, <https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859>
- [34] Wikipedia, http://en.wikipedia.org/wiki/Frigyes_Karinthy
- [35] Tencent QQ Quan, <http://user.qqzone.qq.com/19990210/blog/1332400269>
- [36] Wikipedia, <http://en.wikipedia.org/wiki/Renren>